Simplifying Network Management with Lockdown

Andrew Blaich
University of Notre Dame
Department of Computer Science and Engineering
Notre Dame, IN 46556
ablaich@cse.nd.edu

Qi Liao
University of Notre Dame
Department of Computer Science and Engineering
Notre Dame, IN 46556
qliao@cse.nd.edu

Aaron Striegel
University of Notre Dame
Department of Computer Science and Engineering
Notre Dame, IN 46556
striegel@cse.nd.edu

Douglas Thain*
University of Notre Dame
Department of Computer Science and Engineering
Notre Dame, IN 46556
dthain@cse.nd.edu

ABSTRACT

The administrator of an enterprise network has a responsibility to enforce the policies on the network. Yet, most security mechanisms do not map well to the intended policies. This has been due to the prevalence of simplistic tools that have poor enforcement but, yet are easy to manage. While advanced commercial solutions do exist that have stronger enforcement, they are significantly harder to manage. To that end, we propose Lockdown, a policy-oriented security approach that builds on the concept of local context to deliver a lighter weight approach to enterprise network security while striking a balance between the level of enforcement and level of management available to the network administrator. In this paper, we describe how the Lockdown approach streamlines the process of network security management from network auditing to visualization to policy mapping to enforcement to validation. We demonstrate the strength of Lockdown through detailed assessments of an enterprise university network to show how local context significantly improves network management for the system administrator.

1. INTRODUCTION

Network security policy is complicated and difficult to fully manage in an enterprise setting. While the commercial solutions provide a rich variety of mechanisms, they lack a streamlined approach that would allow them to be setup and managed efficiently. In a recent Computer Crime and Security 2007 survey [25], the collected data showed that the commonly deployed security solutions are the simpler and less effective ones. For instance, the survey showed that the lower end tools which include traditional firewalls have near ubiquitous deployment, found in upwards of 97% of the networks surveyed. The higher end solutions, such as endpoint security client software/(NAC) have considerably less deployment with their deployment in 2007 at 27 % down from 31% in 2006.

In short, there appears to be two broad categories of solutions for network administrators:

1. simple policy enforcement with easier manageability
2. rich policy enforcement with complex management

Clearly as evidenced by [25], network administrators are choosing the first category of tools in favor of manageability

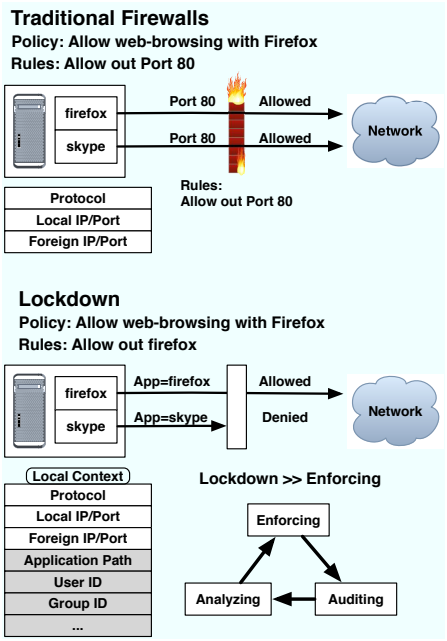

Figure 1: Adding local context to streamline rule creation.

*This work was supported in part by the National Science Foundation through the grants CNS03-47392 and CNS-05-49087.

Symposium on Usable Privacy and Security (SOUPS) 2008, July 23–25, 2008, Pittsburgh, PA USA
.

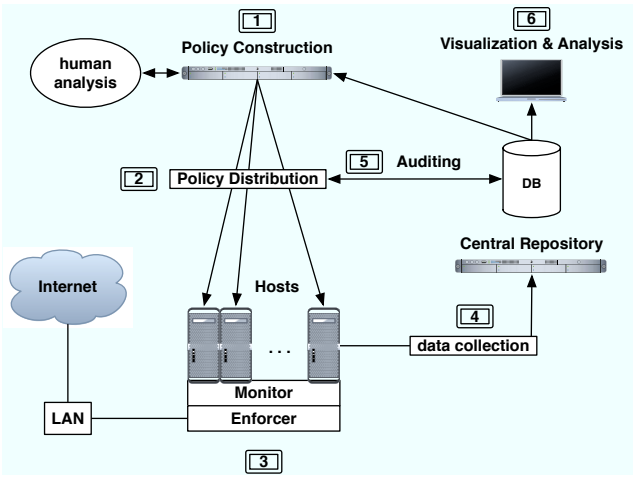

Figure 2: Lockdown System Architecture, the number is indicative of the closed loop Lockdown operates in, i.e. 1,2,...,6,1,2,...,6

over security. Notably, such tools often derive their ease of management by implicitly trusting that Layer 3 (IP) and Layer 4 (Port) map to user and application.

To further illustrate the problem with lower end tools in regards to enforcement, consider an example based on *iptables* as shown in Figure 1. In the example, the desired high level policy is to allow outbound connectivity for non-secure web browsing. The natural rule for *iptables* or any network firewall would be to allow outbound port 80 with state preservation. Critically, the reliance on inference of application from port number allows for other applications not originally intended in the high level policy to gain access out of the network. As a result, the security mechanism of *iptables* would allow not only web browsing but also applications such as Skype or Gnutella that can arbitrarily configure port numbers or tunnel directly over HTTP. Although increasingly popular solutions such as deep packet inspection can address those concerns, those solutions suffer in the presence of user-agent spoofing and altogether fail when faced with end-to-end encryption [2].

We posit that the perspective at which enforcement and monitoring occurs needs to be shifted in such a way that the *local context*, i.e. the why of a connection versus the where, forms the foundation of the security approach. By infusing enforcement with local context, high level policy can be followed more stringently as is seen in the "Lockdown" example of figure 1.

While enforcement with local context is certainly not a new concept, we argue that the monitoring (gathering, auditing, analyzing) of local context has profound implications for network management. Critically, local context strikes the ideal balance between ensuring proper policy enforcement and complexity of mechanism while also improving network management.

To that end, we present the Lockdown security approach in this paper. Lockdown is a streamlined management approach for the enterprise network based on local context. Figure 2 gives an overview of the system architecture. Lockdown has several components that make up the system:

1. *Policy*: Lockdown improves the mapping of policy to mechanism by leveraging local context for rule construction. Local context allows for Lockdown to offer reasonable levels of expression while preserving clear observability from the policy statement which the rules originate from.

2. *Enforcement*: The Enforcer component allows for local context rules to be enforced through the use of a pluggable security module.

3. *Monitor*: The Monitor component enables the Lockdown system to gather local context natively from the hosts and forward it to a central repository.

4. *Auditing*: Auditing is used to validate that policy is being followed.

5. *Analysis & Visualization*: These components allow for easy management of the network through visualization of the connections and discovering chains that occur as a result of user + application interactions. The data collected is in turn analyzed for patterns among resource usage and potential problem areas.

Lockdown provides an economy of mapping mechanism to policy that is expressive, but also lighter weight than high end host-based solutions. The ability to visualize and audit the network creates a platform to explore the local context gathered from all the hosts allowing for validation of policy as well as determining any sort of interesting anomalies occurring among user usage. Finally, the ability to identify problems and tie them to specific application easily are the key features of our system.

Lockdown serves as a robust addition to an enterprise network infrastructure. Critically, we note that the goal of Lockdown is to complement, not replace the security infrastructure by plugging into existing work from data mining, anomaly detection, firewall analysis/management, and policy mapping.

2. ARCHITECTURE

The Lockdown system architecture is a distributed system composed of host installable components in addition to a central repository for the database. Utilizing a distributed data-collection system of monitors and enforcers installed on every host the mapping of mechanism to policy is fully realized via use of the local context. Since the architecture is distributed, a global policy can be created for all hosts and then finely tuned for each host as individual needs may change over time in a closed loop.

2.1 Policy

One of the most critical aspects of any enterprise security approach is how the enterprise policy is mapped to the network security mechanisms, be they end host or in-network mechanisms. In the ideal management case, the configurable aspects of the mechanisms (typically rules) map in an easily observable manner to the policy, be it one or more rules tied to a specific policy statement. The diverse array of work in the area of natural language processing with regards to security [29, 4] is a testament to the appeal of said aspect. In a similar vein, the mechanisms must be able to capture reasonable levels of expressiveness to enable reasonable confidence that the policy item is indeed addressed by the mechanism.

Hence, the wide array of work on formal security expressiveness addresses this need [9, 15, 32].

However, as any administrator or researcher in systems will attest, the practical limits of systems and resources make complexity a natural enemy of robustness and security. We do not focus on the theoretical foundations of policy mapping but rather to focus on how local context offers a compelling economy of expressiveness, i.e. significant improvement to the efficacy of managing the network with negligible increases to complexity. In short, local context adds the ability to create rules that are cognizant of the normal UNIX user (akin to a network ACL) and application (name, path, arguments). In an oversimplified sense, the security mechanism offered by Lockdown would appear to simply be an enhanced firewall, i.e. *iptables++*. In contrast, as we will show in later sections, the consideration of local context can dramatically improve not only policy mapping but also network assessment, mechanism auditing, and debugging, i.e. streamlining the core management aspects expected of a system administrator.

2.2 Enforcer

The job of the Enforcer component is to interpose when a network operation is attempted and either allow the operation to continue or to deny and shutdown the network attempt based on local context. Since sockets are the method in which the kernel manages network connectivity, it is a prime place to enforce on. Also, working at this level in the operating system allows direct access to the local context information. With the Enforcer working at the system call level, it is able to solve the problem of acting on local context, providing interesting feedback to applications that have their sockets denied, and achieve an environment that is easily managed since it has a better view of what is actually occurring with network connectivity. We choose to use Linux Security Modules to build the Enforcer.

The Linux Security Module framework or LSM, standard in the 2.6 Linux kernel, but available as a patch for the 2.4 version, has several hooks placed within an assortment of system calls that allow upcalls to loadable modules implementing the functions [31].To enforce network connectivity we focus on the socket hooks.We can determine whether certain sockets should be created before they are, or if an incoming connection on a listening host should even be accepted. If upon passing through the LSM the firewall or some sort of Intrusion Prevention/Detection System located down the line chooses to close the flow this can still be done since the LSM is an additional security feature and the proper way to secure a system is through a non-homogenous layered approach.

Additionally, the notion of being able to more effectively debug network connectivity issues is greatly enhanced using the LSM for enforcement. Figure 3 presents two cases. In the *iptables* case, all outgoing traffic is blocked as an *java* application we wrote to connect to our laboratory's web-server is initiated. The *java* application had a socket timeout value set to 30 seconds, where the default for *java* sockets is no time-out. Since *iptables* will simply drop packets and not inform the application of what is happening the application sat until 30 seconds had passed and then the user was informed of the *SocketTimeoutException*. Had there not been a timeout value set (default for *java*), the application would have sat indefinitely with no feedback for the user.

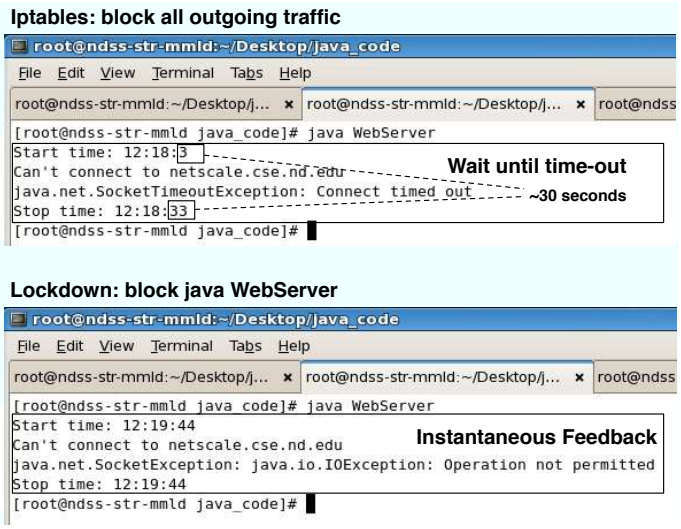

Figure 3: The application view of enforcement comparing *iptables* (above) versus Lockdown (below)

In the Lockdown case, a rule was put in to block the *java WebServer* application. As soon as the application was executed the *IOException* was returned informing the user that the operation was not permitted. By using an LSM based enforcement approach the user/administrator is able to debug the problem easier than if packets were simply being dropped with no feedback on what is occurring.

2.3 Monitor

The Monitor is deployed onto each host possible throughout the network. The purpose of the Monitor is to gather the local context related to network activity on the host. Since the only way to accurately know exactly what is occurring on a host is to monitor it by physically being at the machine, either in software or by some other means, this is why we deploy our monitor in a distributed fashion among all the hosts on the network instead of monitoring a central location. The information gathered from each host allows for accurate auditing and policy management in a closed loop, rather than inferring information and/or requiring all connections to flow through a central entity.

The data collection from April to April 2008, has been using the shell script Monitor. At the time of writing, the agent is deployed on *Linux*, *Solaris*, and *OS X* platforms while a native *Windows* version is under development. The Monitor is a root installed *BASH* shell script that gathers and sends the local context of the monitored host to the central repository. The simplicity of the Monitor lies in its ability to run the commonly found tools *netstat*, *ps*, and *lsof* on any *Linux/Unix/BSD* based operating system.

These three tools are used instead of any single one in order to gather the entire local context which includes: *ppid, pid, uid, gid, foreign ip, foreign port, local ip, local port, full application path + arguments* among other information.

2.4 Visualization & Analyzer

The Visualization and Analysis components are built into a standalone application that enables a network administrator to easily trace through a period of time's connections. The application presents numerous statistics and other use-

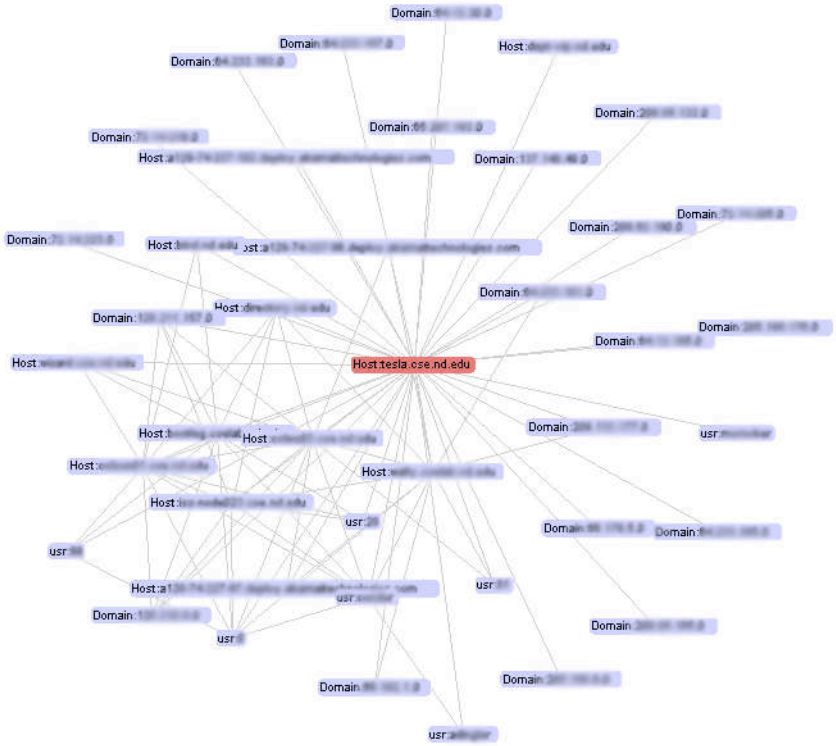

Figure 4: Lockdown Viewer, displaying the connections for a specific host.

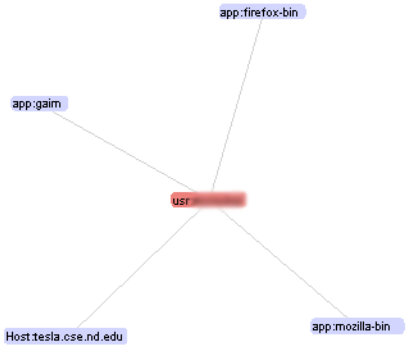

Figure 5: Lockdown Viewer, displaying a specific user discovered using a host.

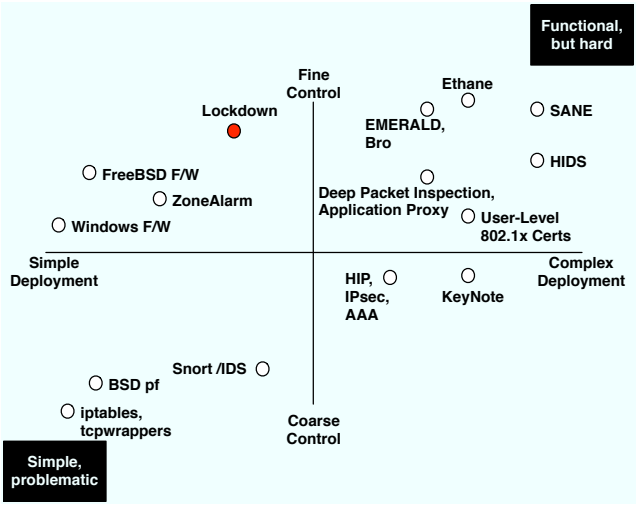

Figure 6: Contextualizing Lockdown within existing network security mechanisms

ful information gleaned from data-mining algorithms that are run. Network connections are visualized via the *prefuse* visualization toolkit, as seen in figures 4 and 5, and the Viewer is able to drill down to different levels of detail simplifying the administrators job of viewing how the network is used and who is using what when. The search and graphing of dependencies among the nodes in the graph most closely resembles the Leslie Graphs as used in [16] except our networking graphing provides a robust level of exploration and analysis on the dependencies among hosts, users, and applications as opposed to just host to host connections.

3. RELATED WORK

In a broad sense, the work in this paper touches on the vast array of research already conducted with regards to firewall/policy analysis [18, 3, 17, 10, 11], intrusion detection [1, 19, 24, 7], user/host authentication [6, 23], and recent clean slate design security efforts [13, 12]. Figure 6 attempts to capture where Lockdown lies on the axes of deployment complexity (x axis) and granularity of control (y axis). In some sense, the figure captures the range of solutions rang-

ing from lightweight, simple firewall solutions to pervasive, heavyweight solutions that encompass the entirety of the enterprise. Notably, the figure focuses on standards body and research works with the primary discussion below regarding commercial solutions.

At the lower end of deployment complexity, traditional host-based and in-network solutions are located including firewalls (*iptables & OpenBSD-pf*) and flow monitoring solutions that act in an application-independent manner (*Snort*, classic anomaly detection systems, etc.). Notably, these devices suffer from the inability to understand the context of the connection, acting on Layer 3 / Layer 4 data or broad patterns of activity (signatures, anomalous traffic patterns, etc.). However, given that these techniques are often the initial mechanism for applying policy, significant research has explored how to validate policy mapping. Hamed and Al-Shaer [18] noted a taxonomy of conflicts in policy for network security devices with their previous work applying graph-based boolean function manipulation to distributed policy analysis [3]. Guttman [17] constructed a global policy definition language with algorithms for verification while Bartal et al in [10] separated policy from topology along with a more modular architecture with Firmato.

Ioannadis et. al in [21] introduced KeyNote which addressed the issue of distributed firewall management. While the kernel enforcement mechanism of KeyNote shares aspects of the Lockdown enforcer component, KeyNote operates largely in the same rule domain as traditional IP firewall rules (IP, port) only with authentication (digital credentials) enforcing user identity. The recent clean slate efforts of SANE [13] and Ethane [12] move enforcement to the network switch itself with Ethane operating in a slightly less heavyweight manner than SANE. Similar to KeyNote, Ethane and SANE force the users through a centralized controller (digital certificates via IKE in KeyNote) to validate connectivity with the resulting authentication being a prerequisite for proper LAN routing. Critically, the clean slate architectures of SANE/Ethane represent a sizeable cost in terms of changing network hardware. Moreover, we note that none of the noted works address management, focusing exclusively on the how of enforcement rather than how management of the network might be improved through visualization or auditing.

Deep packet inspection (DPI), i.e. application-specific proxying, trades processing speed for the ability to fully evaluate the state of the application-layer protocol. A typical deployment of DPI might involve an in-band application-aware IDS or forcing users to authenticate through an application-specific proxy (ex. web proxy). While this is marginally effective for the most basic of applications, DPI must continually react to application protocol enhancements and applications exploiting 'benign' operations to bypass filtering. Moreover, DPI offers little benefit when the traffic itself is encrypted (SSH, SSL, etc.). While work has been conducted on how to infer applications types despite encryption [20], the potential for widespread usage of encryption with IPv6 is problematic.

On the commercial side, numerous solutions exist across the entirety of the deployment complexity spectrum. Enhanced firewalls provide normal firewall rules with additional options for consideration of applications and for detecting changes to the application itself (Windows XP Firewall, ZoneAlarm). Management software such as Microsoft SMS (Server Management System) and others allows for management of the distributed policies. However as noted earlier, these tools can make security-based connectivity issues difficult to detect and offer little in the manner of validation or visualization of the network itself.

Additional to Microsoft SMS there have been several other projects from the MS research group in regards to enterprise network network management. Strider ,[27], discusses the development of a system to monitor the persistent-state interactions on a system in the form of an always-on monitoring agent. The goal of this project was to troubleshoot system mis-configurations and keep state so that a system can be rolled back to a working state. Peer Pressure, [28], uses a statistic model to help diagnose the root-cause of possible misconfigurations on a system in an automated fashion. However, this system reports a large number of false positives for systems which are highly customized by the end-user. The search and graphing of dependencies among the nodes in the graph on Lockdown most closely resembles the Leslie Graphs described in [16]. Lockdown, though, provides a much more robust level of exploration and analysis on the dependencies among hosts, users, and applications as opposed to just host to host connections which appear in [16]. Additionally, Sherlock, [8], presents the use of inference graphs and data collection to detect faulting nodes in a network and dependencies among nodes through the use of an agent loaded onto machines in the network. The Sherlock agent is however, a packet level analyzer and does not achieve the granularity that the Lockdown Monitor does. While Sherlock can help detect performance problems, the reasons for the problems are limited to its granularity of data collection and analysis.

With regards to the high end, we note several prominent solutions including Cisco Security Agents, Endforce, Consentry, Alterpoint, and Elemental Secrity. In a broad sense, the solutions can be divided into three different groups. The first group are based on signature databases whereby application network accesses are analyzed by a host agent and compared to the signature database for possibly exploits and security warnings. The second group employs signature analyses to search for common security holes such as buffer overflows with data logged for future analysis. The third group of solutions such as Cisco's NAC employ a mixture of network-level and host-level control with user authentication to control network security. While these solutions are quite powerful, the pervasive commercial solutions are often time consuming to configure and manage requiring significant IT investments to employ effectively. Hence, as noted in the introduction, deployment of such solutions has largely been limited to a minority of enterprise environments [25].

Finally, we note related work in the area of intrusion detection. Specifically, Lockdown does not attempt to fill the role of a host-based IDS (HIDS) [24, 26, 22]. Notably, several works noted the need for local context [5, 30] for better policing but did not focus on how to gather or analyze the information. Conversely, other IDS works [24, 26, 14] have described approaches for the aggregation of host-based IDS information for centralized analysis. In contrast to the often heavyweight nature of host-based intrusion detection systems and their respective data gathering, Lockdown focuses on maximumizing benefit with minimal cost. As a result, Lockdown trades effectiveness of mechanism for that decision (impact of compromised host) but offers vastly im-

proved management to the network administrator with minimal deployment cost.

4. REFERENCES

[1] Snort: The open source network intrusion detection system. Available at www.snort.org.

[2] 2006.

[3] E. Al-Shaer and H. Hamed. Discovery of policy anomalies in distributed firewalls. In *IEEE INFOCOM*, pages 2605– 2616, Mar. 2004.

[4] E. S. Al-Shaer and H. H. Hamed. Management and translation of filtering security policies. In *IEEE ICC*, 2003.

[5] M. Almgren and U. Lindqvist. Application-integrated data collection for security monitoring. In *RAID '01: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 22–36, London, UK, 2001. Springer-Verlag.

[6] R. Atkinson. IP Authentication Header. *IETF RFC 1826*, Aug. 1995.

[7] S. Axelsson. Intrusion detection systems: A survey and taxonomoy. Technical report, Chalmers University, 2000.

[8] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *Proceedings of ACM SIGCOMM 2007*, 2007.

[9] A. Bandara, A. Kakas, E. Lupu, and A. Russo. Using argumentation logic for firewall policy specification and analysis. In *IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM)*, pages 185–196, Oct. 2006.

[10] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. In *IEEE Symposium on Security and Privacy*, May 1999.

[11] S. Bellovin. Distributed firewalls. *;login*, pages 39–47, November 1999.

[12] M. Casado, M. Freedman, J. Pettit, J.Luo, N. McKeown, , and S. Shenker. Ethane: Taking control of the enterprise. In *SIGCOMM*, 2007.

[13] M. Casado, T. Garfinkel, A. Akella, M. Freedman, D. Boneh, N. McKeown, and S. Shenker. Sane: A protection architecture for enterprise networks. In *15th USENIX Security Symposium*, 2006.

[14] H. Dreger, C. Kreibich, V. Paxson, and R. Sommer. Enhancing the accuracy of network-based intrusion detection with host-based context. In *Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, 2005.

[15] A. El-Atawy, T. Samak, Z. Wali, E. Al-Shaer, S. Li, F. Lin, and C. Pham. An automated framework for validating firewall policy enforcement. In *Proc. of IEEE Workshop on Policies for Distributed Systems and Networks (POLICY'07)*, Bologna, Italy, June 2007.

[16] P. B. et. al. Discovering dependencies for network management. In *Proceedings of workshop on Hot Topics in Networks*, 2006.

[17] J. Guttman. Filtering posture: Local enforcement for global policies. In *IEEE Symposium on Security and Privacy*, May 1997.

[18] H. Hamed and E. Al-Shaer. Taxonomy of conflicts in network security policies. *IEEE Communications Magazine*, pages 134–141, Mar. 2006.

[19] M. Handley, V. Paxson, and C. Kreibich. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Proc. of USENIX Security Symposium*, Aug. 2001.

[20] J. Horton and R. Safavi-Naini. Detecting policy violations through traffic analysis. In *Proceedings of the 22nd Annual Computer Security Applications Conference*, 2006.

[21] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a distributed firewall. In *ACM Conference on Computer and Communications Security*, pages 190–199, 2000.

[22] R. Kemmerer and G. Vigna. Hi-DRA: Intrusion Detection for Internet Security. *IEEE Proceedings*, 93(10):1848–1857, October 2005.

[23] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. *IETF RFC 4423*, May 2006.

[24] P. Porras and P. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proc. of the 20th National Information Security Systems Conference*, pages 353–365, Baltimore, Maryland, Oct. 1997.

[25] R. Richardson. 2007 csi computer crime and security survey. In *The 12th Annual Computer Crime and Security Survey*. Computer Security Institute, 2007.

[26] E. H. Spafford and D. Zamboni. Intrusion detection using autonomous agents. *Comput. Networks*, 34(4):547–570, 2000.

[27] C. Verbowski, E. K. rad Daniels, A. Kumar, Y.-M. Wang, R. Roussev, S. Lu, and J. Lee. Flight data recorder: Always-on tracing and scalable analysis of persistent state interactions to improve systems and security management. In *In Proceedings of Seventh Symposium on Operating Systems Design and Implementation*, 2006.

[28] H. J. Wang, J. Platt, Y. Chen, R. Zhang, and Y.-M. Wang. Automatic misconfiguration troubleshooting with peerpressure. In *in Proc. Sixth Symposium on Operating Systems Design and Implementation (OSDI)*, 2004.

[29] J. Weeds, B. Keller, D. Weir, I. Wakeman, J. Rimmer, and T. Owen. Natural language expression of user policies in pervasive computing environments. In *Proc. of OntoLex 2004 (LREC Workshop on Ontologies and Lexical Resources in Distributed Environments)*, 2004.

[30] M. G. Welz and A. Hutchison. Interfacing trusted applications with intrusion detection systems. In *RAID '01: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 37–53, London, UK, 2001. Springer-Verlag.

[31] C. Wright, C. Cowan, J. Morris, S. Smalley, and G. Kroah-Hartman. Linux security module framework. In *Ottaw Linux Symposium*, 2002.

[32] L. Zhao, A. Shimae, and H. Nagamochi. Linear-tree rule structure for firewall optimization. In *Proceedings of Communications, Internet, and Information Technology*, 2007.